Practica 2.

Lisardo Gayán Tremps
José Luis Melo

11 June, 2019

Contents

1 - Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

El dataset de Titanic: Machine Learning from Disaster se registran los datos de los pasajeros del famoso trasatlántico y se utiliza para predecir los supervivientes. Los datos estan divididos en dos dataset, uno de test y otro entrenamiento, para la creación de modelos de predicción.

2 - Integración y selección de los datos de interés a analizar

Se importan los datos. Primero el dataset train.

```
datostrain <- read.csv("./data/train.csv", stringsAsFactors = FALSE, na.strings = c("NA", ""))
str(datostrain)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  NA "C85" NA "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

Se observa como consta de 891 muestras y 12 variables, entre ellas Survived.

Posteiormente el dataset test.

```
datostest <- read.csv ("./data/test.csv", stringsAsFactors = FALSE, na.strings = c("NA", ""))
str(datostest)
```

```
## 'data.frame':    418 obs. of  11 variables:
##  $ PassengerId: int  892 893 894 895 896 897 898 899 900 901 ...
##  $ Pclass     : int  3 3 2 3 3 3 3 2 3 3 ...
##  $ Name       : chr  "Kelly, Mr. James" "Wilkes, Mrs. James (Ellen Needs)" "Myles, Mr. Thomas Francis
##  $ Sex        : chr  "male" "female" "male" "male" ...
##  $ Age        : num  34.5 47 62 27 22 14 30 26 18 21 ...
##  $ SibSp      : int  0 1 0 0 1 0 0 1 0 2 ...
##  $ Parch      : int  0 0 0 0 1 0 0 1 0 0 ...
##  $ Ticket     : chr  "330911" "363272" "240276" "315154" ...
##  $ Fare       : num  7.83 7 9.69 8.66 12.29 ...
##  $ Cabin      : chr  NA NA NA NA ...
##  $ Embarked   : chr  "Q" "S" "Q" "S" ...
```

Se observa como tiene 418 muestra, y 11 variables. La variable Survived no aparece porque es la que se tiene que predecir.

A continuacion, a la hora de fusionar los datos caben dos posibilidades, asignar "NA" a la variable datostest$Survived o no considerar los datos de survived en train. Se importan, fusionan los datos y se revisa la estructura inicial de los datos.

```
datostest$Survived <- NA
datos <- rbind(datostrain, datostest)
str(datos)
```

```
## 'data.frame':    1309 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  NA "C85" NA "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

A continuación comprobamos los datos que faltan

```
# Busco primero qué variables tienen valores perdidos
missing_numbers <- sapply(datos, function(x) {sum(is.na(x))})
kable(data.frame(Variables = names(missing_numbers), Datos_faltantes= as.vector(missing_numbers))) %>%
  kable_styling(bootstrap_options = "striped", full_width = F, position = "left")
```

Variables	Datos_faltantes
PassengerId	0
Survived	418
Pclass	0
Name	0
Sex	0
Age	263
SibSp	0
Parch	0
Ticket	0
Fare	1
Cabin	1014
Embarked	2

Podemos observar, que en Survived, salen los 418, que tenemos que predecir, por lo que todos los valores de train están informados.

A continuación se detallan las variables y su tipo inicial, este último, se modificara para su mejor analisis.

```
# datostrain1 <- datostrain[,-2]
# data <- rbind(datostrain1, datostest) # Fusion datasets
data <- datos[,-2]
str(data)
```

```
## 'data.frame':    1309 obs. of  11 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
```

```
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  NA "C85" NA "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

```
tipos <- sapply(data, class)
kable(data.frame(Variables = names(tipos), Tipo_Variable= as.vector(tipos))) %>%
  kable_styling(bootstrap_options = "striped", full_width = F, position = "left")
```

Variables	Tipo_Variable
PassengerId	integer
Pclass	integer
Name	character
Sex	character
Age	numeric
SibSp	integer
Parch	integer
Ticket	character
Fare	numeric
Cabin	character
Embarked	character

Las variables, que no tienen datos faltantes, class y sex, se convertiran a factor. La variable cabin tiene muchos datos faltantes, así que en un primer momento no se utilizará.

```
#data$Age <- as.integer(data$Age)
data$Pclass <- as.factor(data$Pclass)
data$Sex <- as.factor(data$Sex)
#data$Embarked <- as.factor(data$Embarked)
#data$Cabin <- as.factor(data$Cabin)

tipos_new <- sapply(data, class)
kable(data.frame(Variables = names(tipos_new), Tipo_Variable= as.vector(tipos_new))) %>%
  kable_styling(bootstrap_options = "striped", full_width = F, position = "left")
```

Variables	Tipo_Variable
PassengerId	integer
Pclass	factor
Name	character
Sex	factor
Age	numeric
SibSp	integer
Parch	integer
Ticket	character
Fare	numeric
Cabin	character
Embarked	character

Una vez modificadas los tipos de valores se resume que:

```
summary(data)
```

```
##   PassengerId   Pclass      Name               Sex           Age
##  Min.   :   1   1:323   Length:1309        female:466   Min.   : 0.17
```

```
##  1st Qu.: 328   2:277   Class :character   male  :843   1st Qu.:21.00
##  Median : 655   3:709   Mode  :character                Median :28.00
##  Mean   : 655                                          Mean   :29.88
##  3rd Qu.: 982                                          3rd Qu.:39.00
##  Max.   :1309                                          Max.   :80.00
##                                                        NA's   :263
##      SibSp            Parch          Ticket              Fare
##  Min.   :0.0000   Min.   :0.000   Length:1309        Min.   :  0.000
##  1st Qu.:0.0000   1st Qu.:0.000   Class :character   1st Qu.:  7.896
##  Median :0.0000   Median :0.000   Mode  :character   Median : 14.454
##  Mean   :0.4989   Mean   :0.385                      Mean   : 33.295
##  3rd Qu.:1.0000   3rd Qu.:0.000                      3rd Qu.: 31.275
##  Max.   :8.0000   Max.   :9.000                      Max.   :512.329
##                                                      NA's   :1
##     Cabin             Embarked
##  Length:1309        Length:1309
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

PassengerId: Variable de tipo entero que contiene el id del pasajero, no existen valores nulos o perdidos.
Pclass: Variable de tipo factor con la categoria asignada al pasajero, no existen valores nulos o perdidos.
Name: Variable de tipo texto con el nombre del pasajero, no existen valores nulos o perdidos.
Sex: Variable de tipo factor con el genero del pasajero (másculino, femenino), no existen valores nulos o perdidos.
Age: Variable de tipo numérico que especifica la edad del pasajero, **existen 263 valores nulos.**
SibSp: Variable de tipo entero que especifica el numero de hermanos/esposa abordo, no existen valores nulos o perdidos.
Parch: Variable de tipo entero que especifica el numero de padres/hijos abordo, no existen valores nulos o perdidos.
Ticket: Variable de tipo texto que indica el numero de ticket, no existen valores nulos o perdidos.
Fare: Variable de tipo numero que especifica la tarifa pagada, **existe 1 valor nulo.**
Cabin: Variable de tipo factor donde se especifica la cabina asignada, **existen 1014 valores perdidos.**
Embarked: Variable de tipo factor que indica el puerto de embarque, **existen 2 valores perdidos.**

3 - Limpieza de datos

3.1. ¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?

Volvemos a mostrar los datos que contienen ceros o elementos vacíos.

```
# Busco primero qué variables tienen valores perdidos
missing_numbers <- sapply(datos, function(x) {sum(is.na(x))})
kable(data.frame(Variables = names(missing_numbers),
                 Datos_faltantes= as.vector(missing_numbers))) %>%
  kable_styling(bootstrap_options = "striped", full_width = F, position = "left")
```

Variables	Datos_faltantes
PassengerId	0
Survived	418
Pclass	0
Name	0
Sex	0
Age	263
SibSp	0
Parch	0
Ticket	0
Fare	1
Cabin	1014
Embarked	2

De las variables existentes a continuación se espedifican aquellas que contienen valores perdido o nulos.

- Age: *existen 263 valores nulos.*

Para imputar valores de **edad**, hay varias opciones desde la más sencilla que sería asignar la media a otras opciones como la propuesta en http://jstatsoft.org/article/view/v045i03 Data Analysis with R, A comprehensive guide to manipulating, analyzing and visualizing data in R Pag 373 - 386. Nosotros imputaremos los valores de edad por la media de

Comprobamos la variable Age

```
summary(data$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.17   21.00   28.00   29.88   39.00   80.00     263
```

Se comprueba como hay 263 valores nulos. Para asignar el valor de la edad aplicamos el algoritmo rpart, que es un árbol de regresión.

```
# Referencia:
# https://www.rdocumentation.org/packages/rpart/versions/4.1-15/topics/rpart

age_model <- rpart(Age ~ Pclass + Sex + SibSp + Parch + Fare + Embarked,
                       data = data[!is.na(data$Age),], method = "anova")

data$Age[is.na(data$Age)] <- predict(age_model, data[is.na(data$Age),])
summary(data$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.17   22.00   27.43   29.63   37.00   80.00
```

También se podrían imputar por otros métodos, como la media, mediana o mice en lugar del usado rpart.

- Fare: *existe 1 valor nulo.*
  Para imputar valores **Fare**
  Dado que unicamente hay un valor perdido, es posible imputarlo por la media o la mediana en base al puerto de embarque "S" y la clase "3"

```r
data[is.na(data$Fare),]
```

```
##      PassengerId Pclass               Name  Sex  Age SibSp Parch Ticket
## 1044        1044      3 Storey, Mr. Thomas male 60.5     0     0   3701
##      Fare Cabin Embarked
## 1044   NA  <NA>        S
```

```r
M_fare<- subset(data,data$Pclass == '3' & data$Embarked == 'S')
mean(M_fare$Fare, na.rm = T)
```

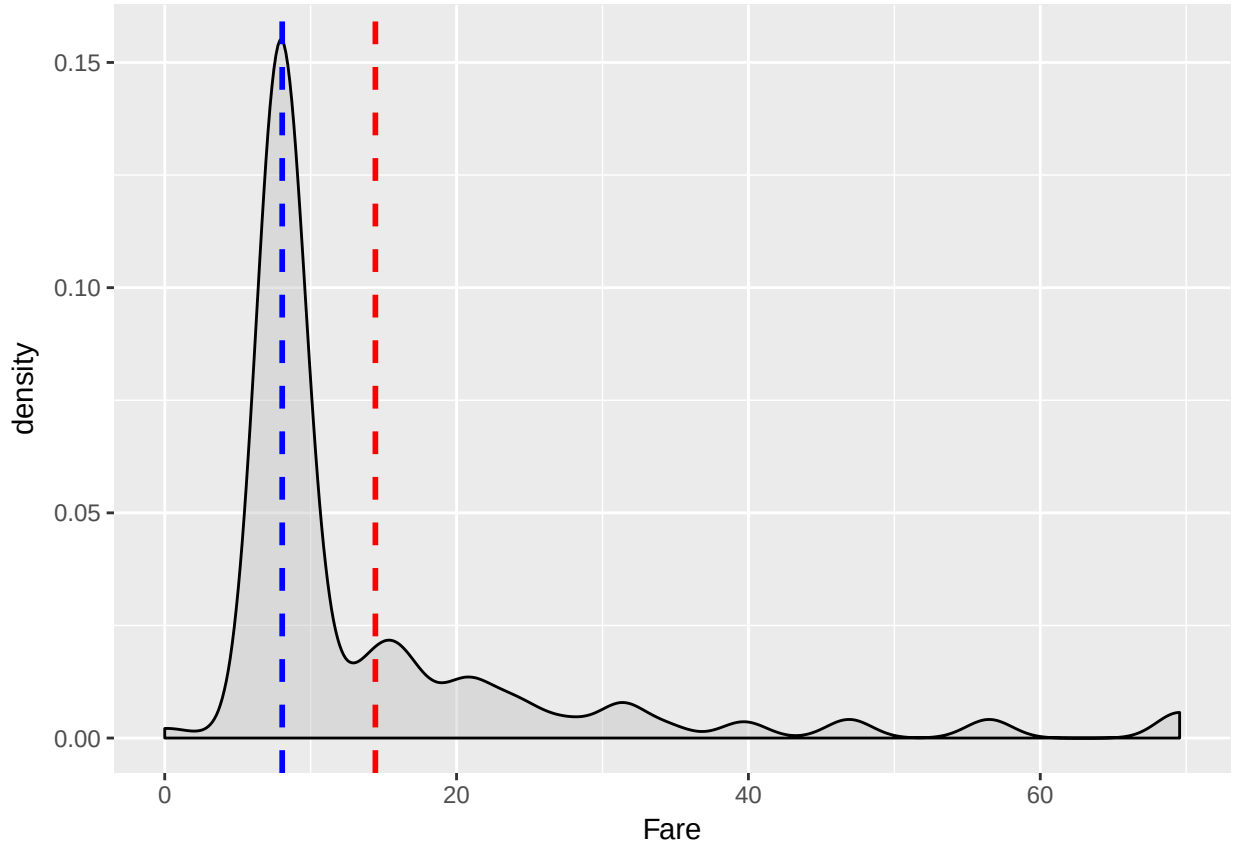
```
## [1] 14.43542
```

```r
median(M_fare$Fare, na.rm = T)
```

```
## [1] 8.05
```

```r
ggplot(M_fare,  aes(x = Fare)) +
  geom_density(fill = 'grey', alpha=0.4) +
  geom_vline(aes(xintercept=median(Fare, na.rm=T)),
    colour='blue', linetype='dashed', lwd=1) +
  geom_vline(aes(xintercept=mean(Fare, na.rm=T)),
  colour='red', linetype='dashed', lwd=1)
```

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```



Observamos como al realizar la gráfica nos dice que hay un valor nulo.

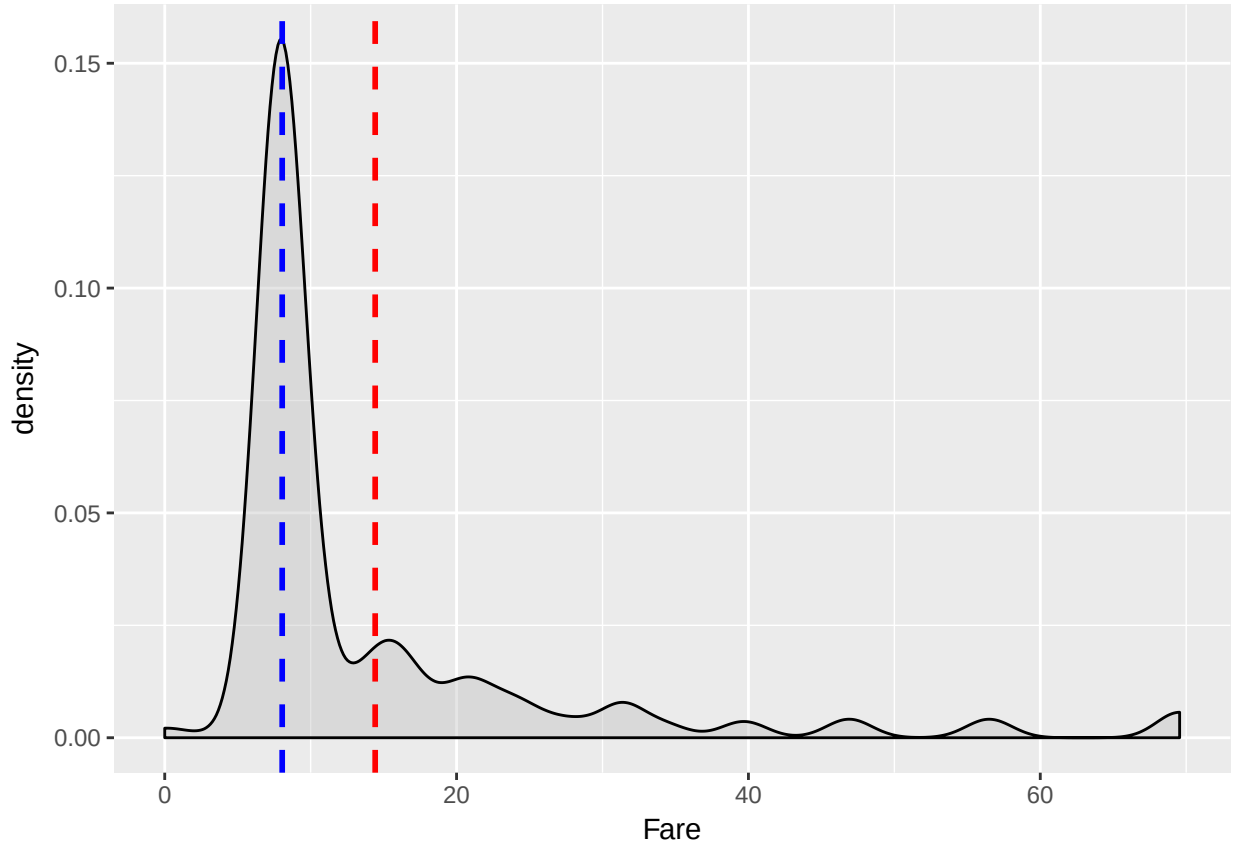
La tarifa de 8.05 coincide con la mediana de los pasajeros de tercera clase que embarcaron en S, por lo que se podría imputar este valor.

```
data$Fare[c(1044)] <- 8.05
data[1044,]
```

```
##      PassengerId Pclass               Name  Sex  Age SibSp Parch Ticket
## 1044        1044      3 Storey, Mr. Thomas male 60.5     0     0   3701
##      Fare Cabin Embarked
## 1044 8.05  <NA>        S
```

Volviendo a representar

```
M_fare<- subset(data,data$Pclass == '3' & data$Embarked == 'S')
ggplot(M_fare,  aes(x = Fare)) +
  geom_density(fill = 'grey', alpha=0.4) +
  geom_vline(aes(xintercept=median(Fare, na.rm=T)),
    colour='blue', linetype='dashed', lwd=1) +
  geom_vline(aes(xintercept=mean(Fare, na.rm=T)),
  colour='red', linetype='dashed', lwd=1)
```



Ahora ya no sale que haya un valor nulo en fare.

- Cabin: *existen 1014 valores perdidos.*
  Para imputar valores **Cabin**

Esta variable tiene muchos valores perdidos, se podria conseguir predecir la cubierta asignada al pasajero pero es un dato que poco beneficio podría traer ya que se puede realizar el analisis con la combinación entre la tarifa y la clase del pasajero.

- Embarked: *existen 2 valores perdidos.*
  Para imputar valores **Embarked**

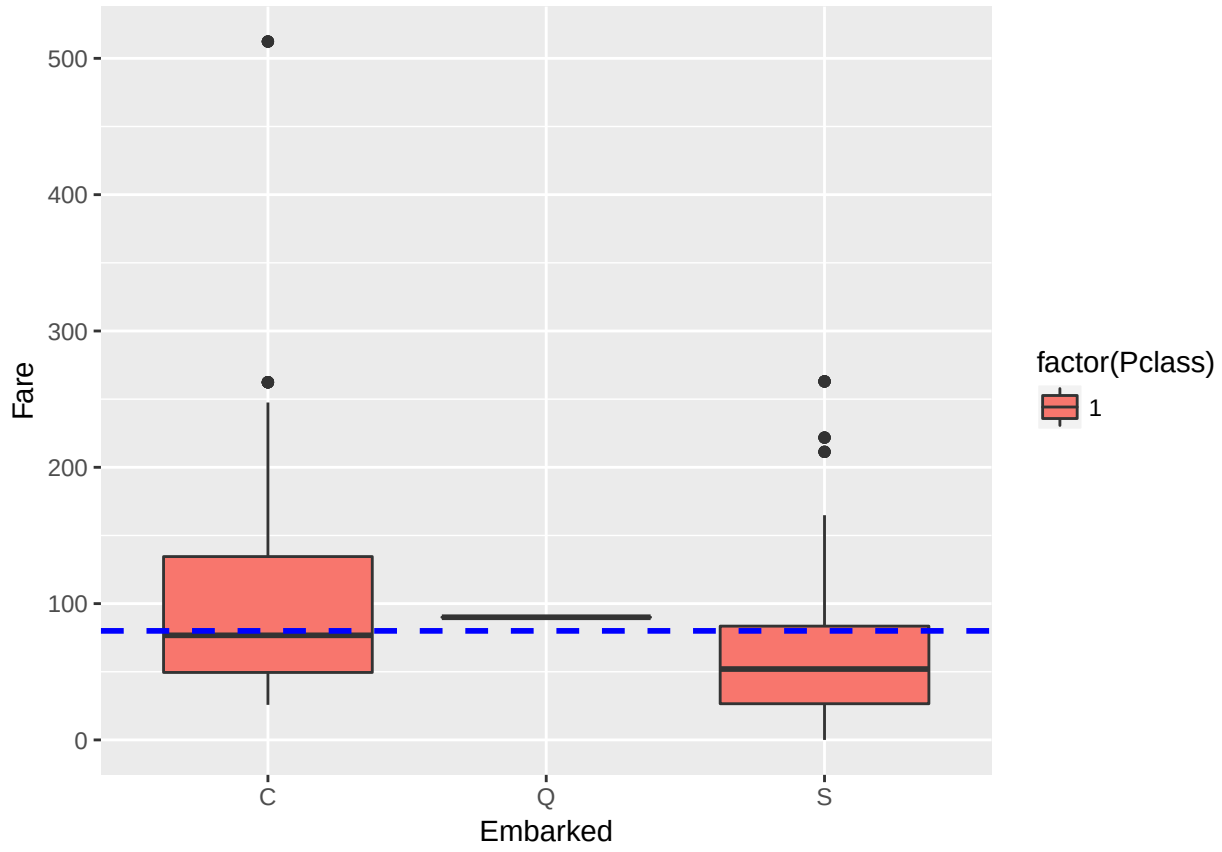
Mostramos los valores perdidos

```
data[is.na(data$Embarked),]
```

```
##     PassengerId Pclass                                         Name    Sex
## 62           62      1                          Icard, Miss. Amelie female
## 830         830      1 Stone, Mrs. George Nelson (Martha Evelyn) female
##     Age SibSp Parch Ticket Fare Cabin Embarked
## 62   38     0     0 113572   80   B28     <NA>
## 830  62     0     0 113572   80   B28     <NA>
```

Al ser unicamente dos valores perdidos, se podría sustituir los valores por la media, en base a otros pasajeros de la misma clase y tarifa (Fare). Los pasajeros han pagado una tarifa de 80 y pertenecian a primera clase.

```
embarked_pass_1 <- data %>%
  filter(PassengerId != 62 & PassengerId != 830 & Pclass == 1)
ggplot(embarked_pass_1, aes(x = Embarked, y = Fare, fill = factor(Pclass))) +
  geom_boxplot() +
  geom_hline(aes(yintercept=80),
    colour='blue', linetype='dashed', lwd=1)
```

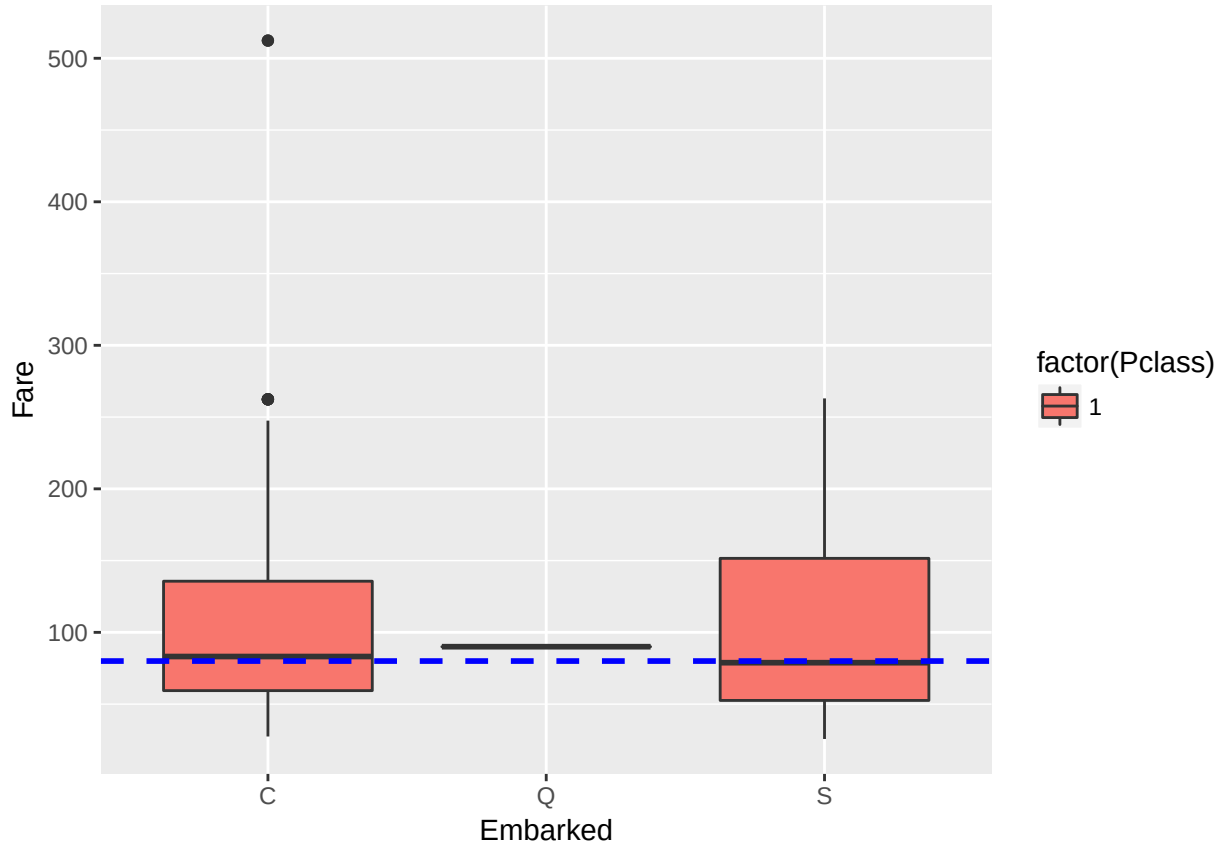


La tarifa de 80 coincide con la media de los pasajeros de primera clase que embarcaron en C, por lo que se podría imputar este puerto.

```
data$Embarked[c(62, 830)] <- 'C'
```

Otra opción, sería considerar también el sexo, ya que principios del siglo XX, no se caracterizaba por una igualdad de hombres y mujeres.

```
embarked_pass_2 <- data %>%
  filter(PassengerId != 62 & PassengerId != 830 & Pclass == 1 & Sex == "female")
ggplot(embarked_pass_2, aes(x = Embarked, y = Fare, fill = factor(Pclass))) +
  geom_boxplot() +
  geom_hline(aes(yintercept=80),
    colour='blue', linetype='dashed', lwd=1)
```



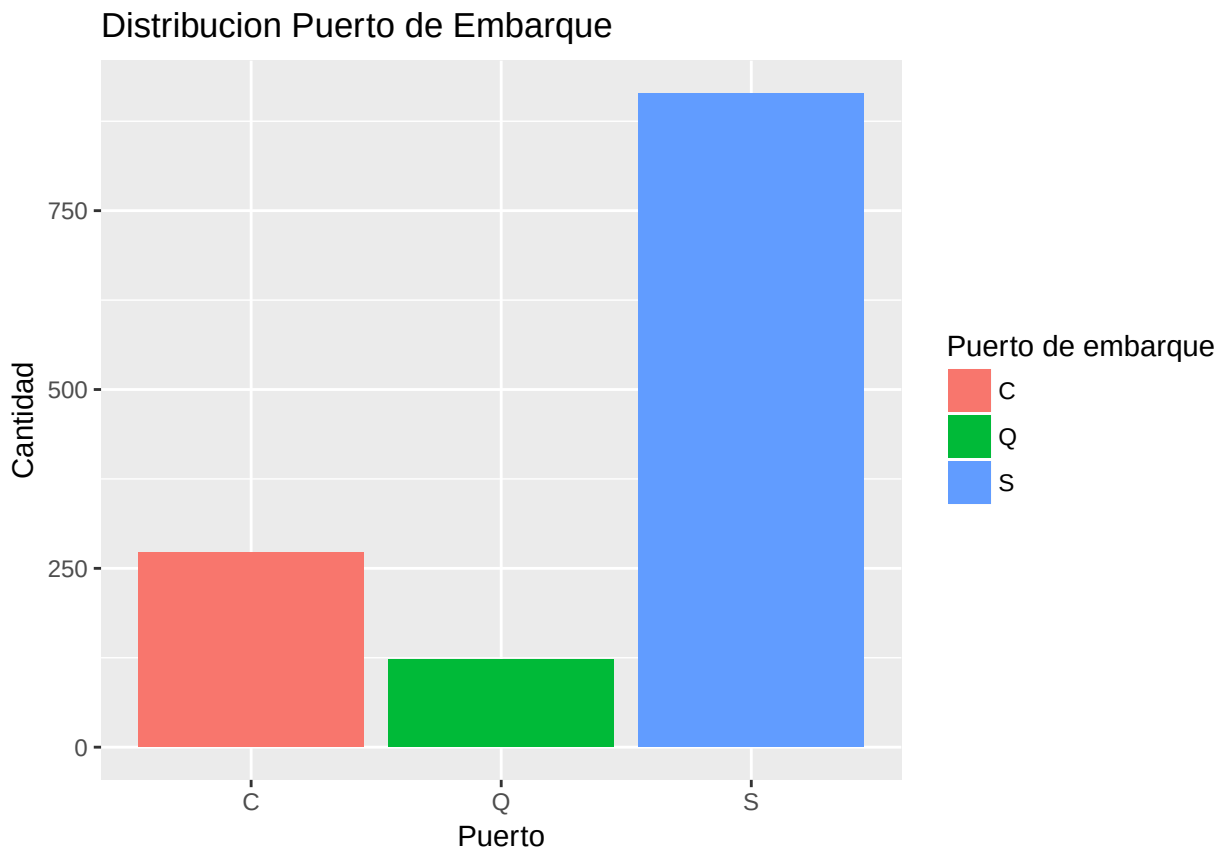
En este caso cualquiera de los 3 puertos tendría una media cercana a 80. Como no creemos que el puerto de embarque este correlacionado con la supervivencia, podríamos dejar cualquiera.

Otra forma de asignar el valor de los puertos de embarque sería asignar el valor más frequente. Calculamos cuantas veces aparece cada puerto de embarque.

```
table(data$Embarked)
```

```
##
##   C   Q   S
## 272 123 914
```

```
qplot(Embarked, data = data,  fill= Embarked) +
  labs (title = "Distribucion Puerto de Embarque",
        x= "Puerto", y = "Cantidad", fill = "Puerto de embarque")
```



Se muestra como el puerto con mayor frequencia es S, así que finalmente asignaremos este valor.

```
data$Embarked[c(62, 830)] <- 'S'
```

A continuación trataremos la variable Name, para crear una variable Title, que nos aporte algo de información.

A partir de Name obtenemos el título de los nombres

```
# Mostramos algunos nombres
head(data$Name)
```

```
## [1] "Braund, Mr. Owen Harris"
## [2] "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
## [3] "Heikkinen, Miss. Laina"
## [4] "Futrelle, Mrs. Jacques Heath (Lily May Peel)"
## [5] "Allen, Mr. William Henry"
## [6] "Moran, Mr. James"
```

Podemos clasificar a las personas por su título.

```
# Grab passenger title from passenger name
data$Title <- gsub("^.*, (.*?)\\..*$", "\\1", data$Name)
```

Mostramos los títulos de las personas que viajaban en el Titanic

```
#Mostramos la variable Title en función de sex
kable(table(data$Title, data$Sex)) %>%
  kable_styling(bootstrap_options = "striped", full_width = F, position = "left")
```

	female	male
Capt	0	1
Col	0	4
Don	0	1
Dona	1	0
Dr	1	7
Jonkheer	0	1
Lady	1	0
Major	0	2
Master	0	61
Miss	260	0
Mlle	2	0
Mme	1	0
Mr	0	757
Mrs	197	0
Ms	2	0
Rev	0	8
Sir	0	1
the Countess	1	0

Podemos reasignar las "Mlle" (Madmoiselle) y "Ms" a "Miss", señoritas y la "Mme" (Madame) a "Mrs" (señora). Las ocurrencias con poca frecuencia las podemos agrupar en otros "Other".

```
# Reagrupando
data$Title[data$Title == 'Mlle' | data$Title == 'Ms'] <- 'Miss'
data$Title[data$Title == 'Mme']  <- 'Mrs'
Other <- c('Dona', 'Dr', 'Lady', 'the Countess','Capt', 'Col', 'Don', 'Jonkheer', 'Major', 'Rev', 'Sir')
data$Title[data$Title %in% Other]  <- 'Other'
```

Volviendo a representarlos en función del sexo.

```
# Mostramos el título en función del sexo
kable(table(data$Title, data$Sex)) %>%
  kable_styling(bootstrap_options = "striped", full_width = F, position = "left")
```

	female	male
Master	0	61
Miss	264	0
Mr	0	757
Mrs	198	0
Other	4	25

La variable Age se podría discretizar y a partir de SibSp y Parch se podría crear una variable con el tamaño de la familia, pero intentaremos un modelo más sencillo, con las variables tratadas hasta ahora.

Una vez añadidos los valores faltantes, se convierte la variable Embarked, Title a factor.

```
data$Embarked <- as.factor(data$Embarked)
data$Title <- as.factor(data$Title)
```

Vuelvemos a añadir la variable Survived y la convertimos a factor.

```
data$Survived <- datos$Survived
data$Survived <- as.factor(data$Survived)
```

Volvemos a mostrar la tabla con el número de valores faltantes. Recordar que en data, no están los valores de la variable Survived.

```
# Busco primero qué variables tienen valores perdidos
missing_numbers <- sapply(data, function(x) {sum(is.na(x))})
kable(data.frame(Variables = names(missing_numbers),
                 Datos_faltantes= as.vector(missing_numbers))) %>%
  kable_styling(bootstrap_options = "striped",
                full_width = F, position = "left")
```

Variables	Datos_faltantes
PassengerId	0
Pclass	0
Name	0
Sex	0
Age	0
SibSp	0
Parch	0
Ticket	0
Fare	0
Cabin	1014
Embarked	0
Title	0
Survived	418

Representamos la tabla con los tipos de los valores.

```
tipos_new <- sapply(data, class)
kable(data.frame(Variables = names(tipos_new), Tipo_Variable= as.vector(tipos_new))) %>%
  kable_styling(bootstrap_options = "striped", full_width = F, position = "left")
```

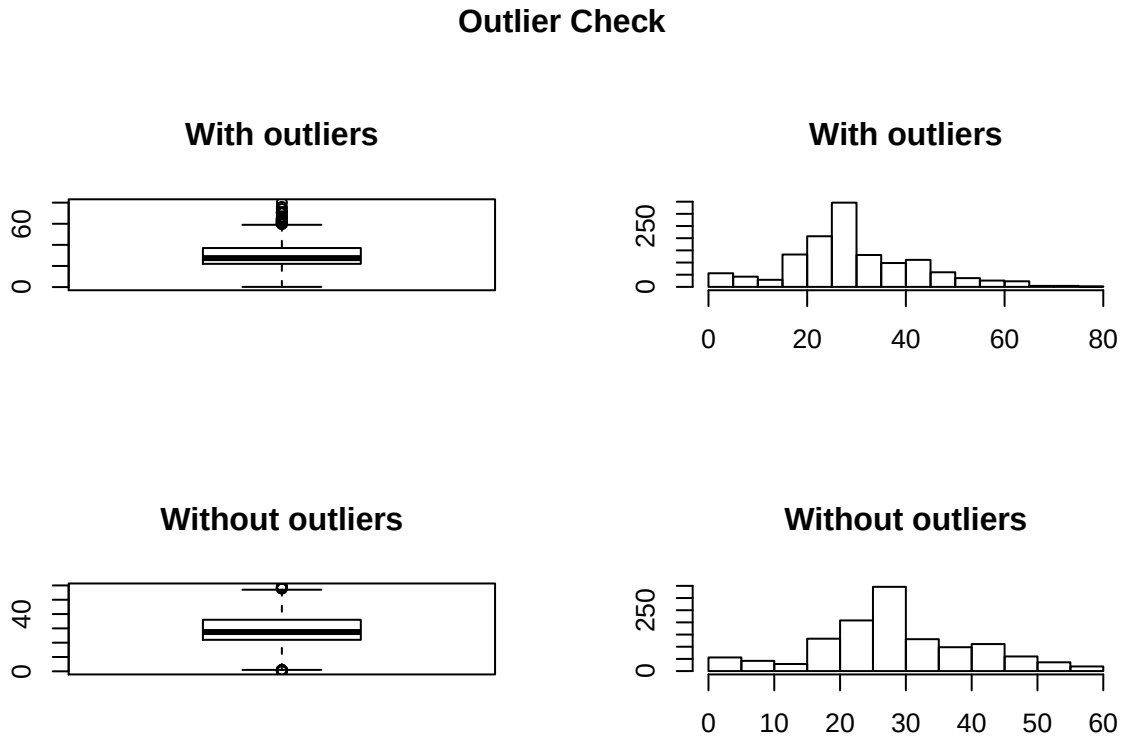
Variables	Tipo_Variable
PassengerId	integer
Pclass	factor
Name	character
Sex	factor
Age	numeric
SibSp	integer
Parch	integer
Ticket	character
Fare	numeric
Cabin	character
Embarked	factor
Title	factor
Survived	factor

3.2. Identificación y tratamiento de valores extremos.

Los valores extremos tendrían sentido en los campos Fare y Age. Procedemos al análisis de los valores extremos representando Fare y Age con los valores extremos y sin ellos.

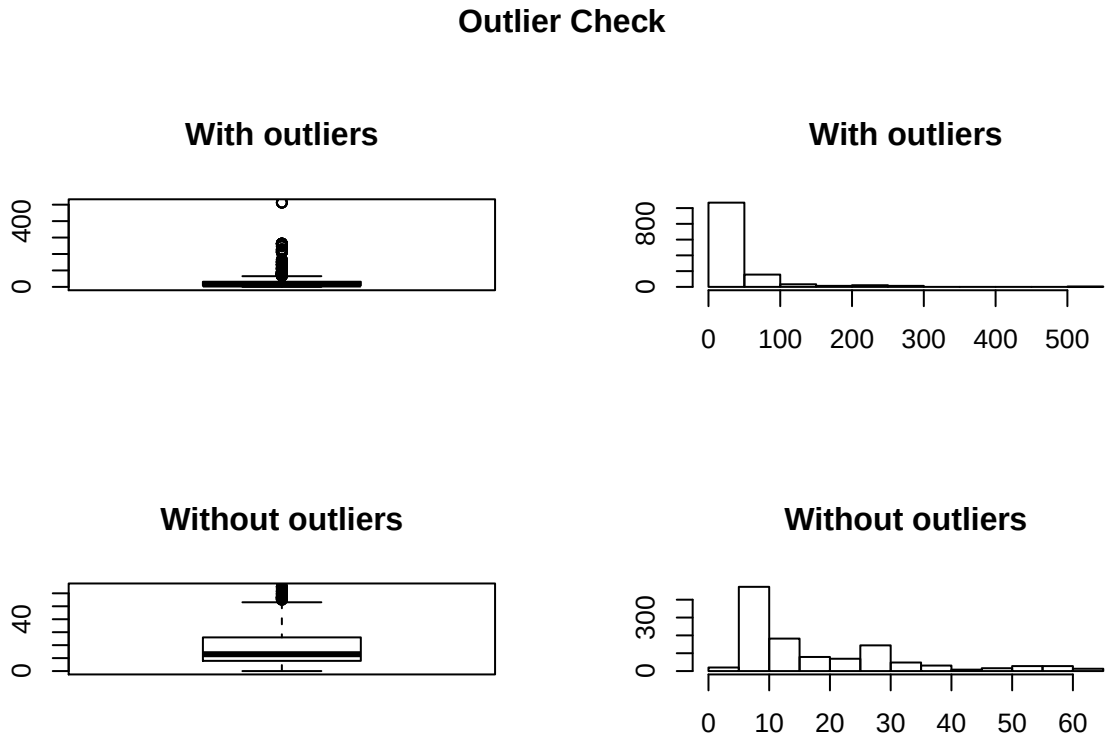
```
# Referencia:
# https://www.r-bloggers.com/identify-describe-plot-and-remove-the-outliers-from-the-dataset/
outlierKD <- function(dt, var) {
     var_name <- eval(substitute(var),eval(dt))
     na1 <- sum(is.na(var_name))
     m1 <- mean(var_name, na.rm = T)
     par(mfrow=c(2, 2), oma=c(0,0,3,0))
     boxplot(var_name, main="With outliers")
     hist(var_name, main="With outliers", xlab=NA, ylab=NA)
     outlier <- boxplot.stats(var_name)$out
     mo <- mean(outlier)
     var_name <- ifelse(var_name %in% outlier, NA, var_name)
     boxplot(var_name, main="Without outliers")
     hist(var_name, main="Without outliers", xlab=NA, ylab=NA)
     title("Outlier Check", outer=TRUE)
     na2 <- sum(is.na(var_name))
     cat("Outliers identified:", na2 - na1, "n")
     cat("Propotion (%) of outliers:", round((na2 - na1) / sum(!is.na(var_name))*100, 1), "n")
     cat("Mean of the outliers:", round(mo, 2), "n")
     m2 <- mean(var_name, na.rm = T)
     ###
     # cat("Mean without removing outliers:", round(m1, 2), "n")
     # cat("Mean if we remove outliers:", round(m2, 2), "n")
     # response <- readline(prompt="Do you want to remove outliers and to replace with NA? [yes/no]: ")
     # if(response == "y" | response == "yes"){
     #      dt[as.character(substitute(var))] <- invisible(var_name)
     #      assign(as.character(as.list(match.call())$dt), dt, envir = .GlobalEnv)
     #      cat("Outliers successfully removed", "n")
     #      return(invisible(dt))
     # } else{
     #      cat("Nothing changed", "n")
     #      return(invisible(var_name))
     # }
     ###
}
```

```
outlierKD(data, Age)
```




```
## Outliers identified: 40 nPropotion (%) of outliers: 3.2 nMean of the outliers: 64.45 n
```

```
outlierKD(data, Fare)
```




```
## Outliers identified: 171 nPropotion (%) of outliers: 15 nMean of the outliers: 135.25 n
```

Como son perfectamente aceptables las edades y que haya gente que pagara mucho más por su billete, al ser el primer viaje del transatlántico más grande de la epoca, decidimos no cambiar ningún valor.

4. Análisis de los datos.

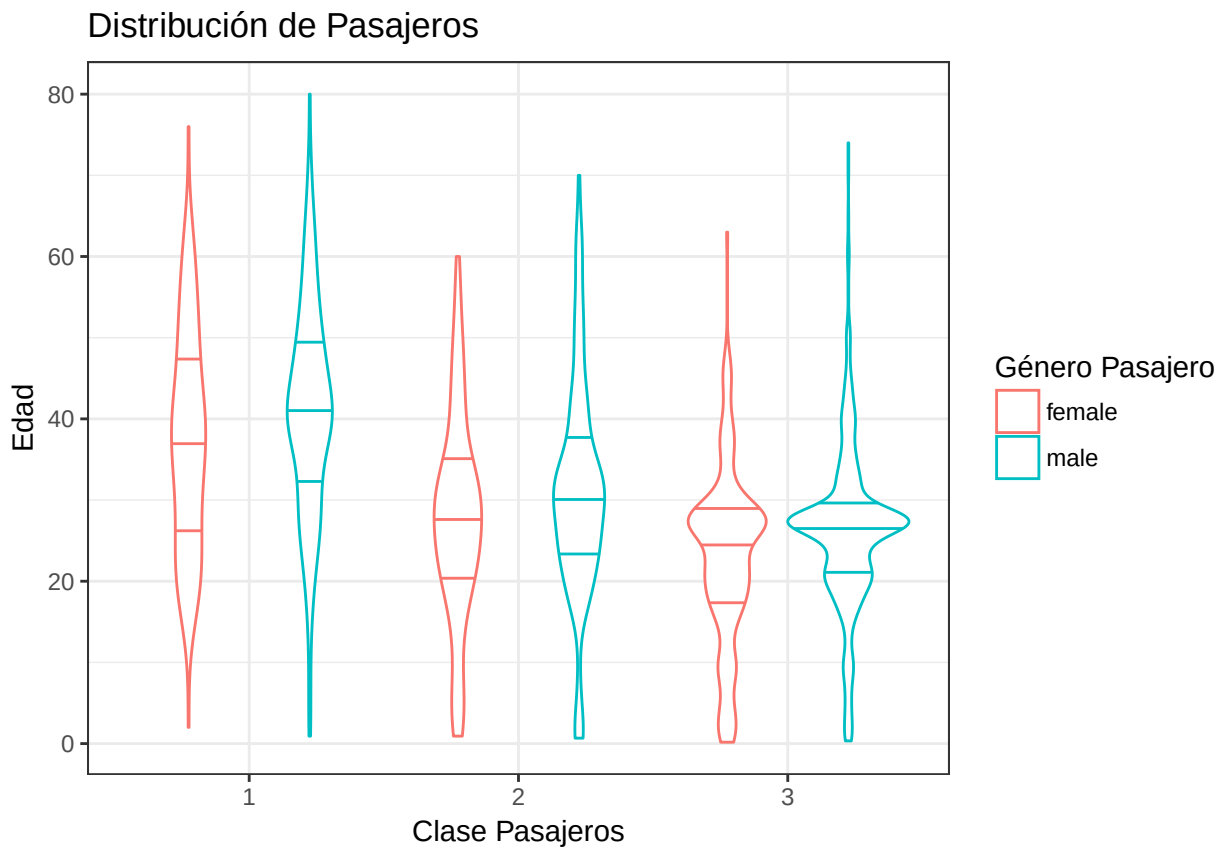
4.1. Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar)

- Análisis estadístico descriptivo.

Se utilizarán las variables de Edad, Clase, Género y Puerto de Embarque para realizar el análisis de los datos.

La distribución de los pasajeros según indica el siguiente grafico.

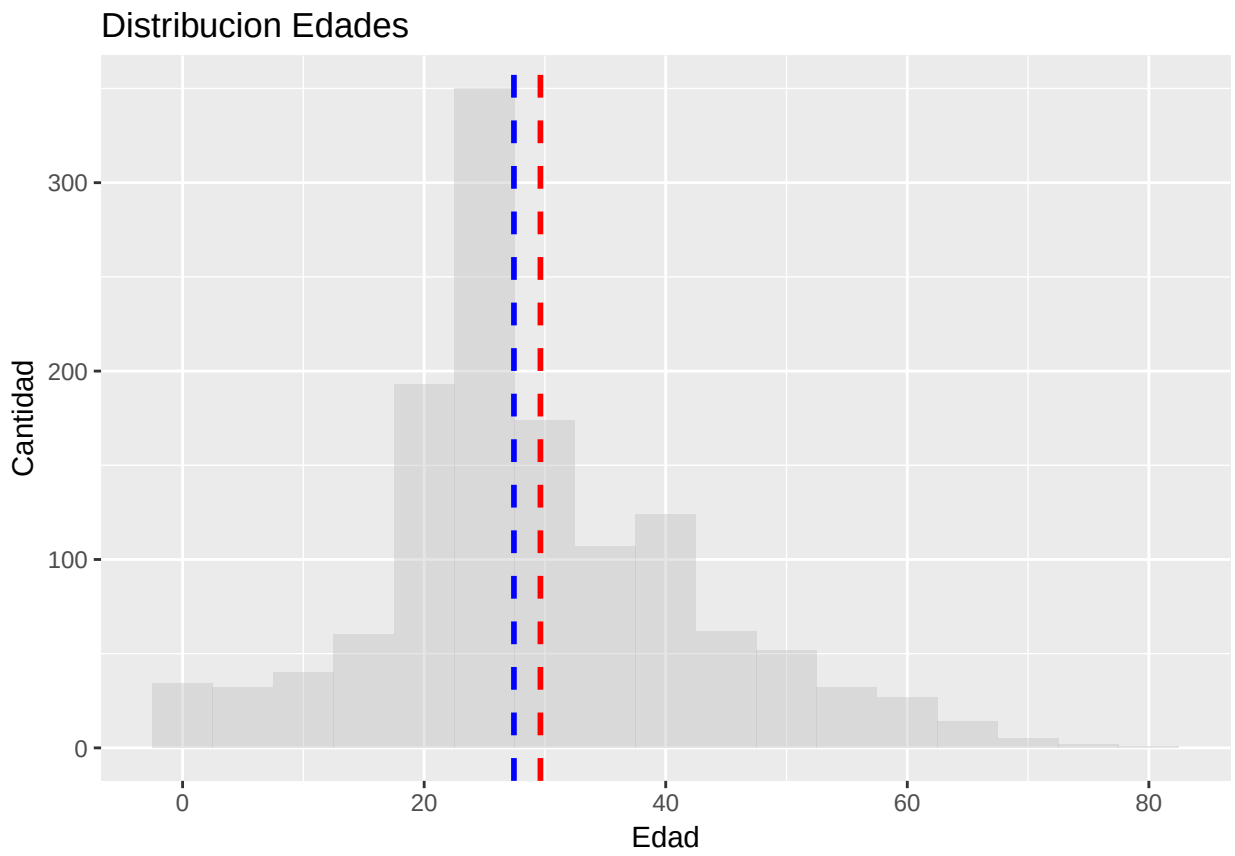
```
ggplot(data = data) +
  geom_violin (aes(Pclass, Age, colour = factor(Sex)),draw_quantiles = c(0.25, 0.5, 0.75)) +
  labs (title = "Distribución de Pasajeros", x= "Clase Pasajeros",
        y = "Edad" , colour = "Género Pasajero") +
  theme_bw()
```



Representando la edad de los pasajeros obtenemos:

```
#Test_age <- na.omit(data$Age)
#Test_age <- as.integer(Test_age)


ggplot(data,  aes(x = Age)) +
  geom_histogram(binwidth = 5, fill = 'grey', alpha=0.4) +
  geom_vline(aes(xintercept=median(Age, na.rm=T)),
    colour='blue', linetype='dashed', lwd=1) +
  geom_vline(aes(xintercept=mean(Age, na.rm=T)),
  colour='red', linetype='dashed', lwd=1) +
  labs (title = "Distribucion Edades", x= "Edad", y = "Cantidad" )
```



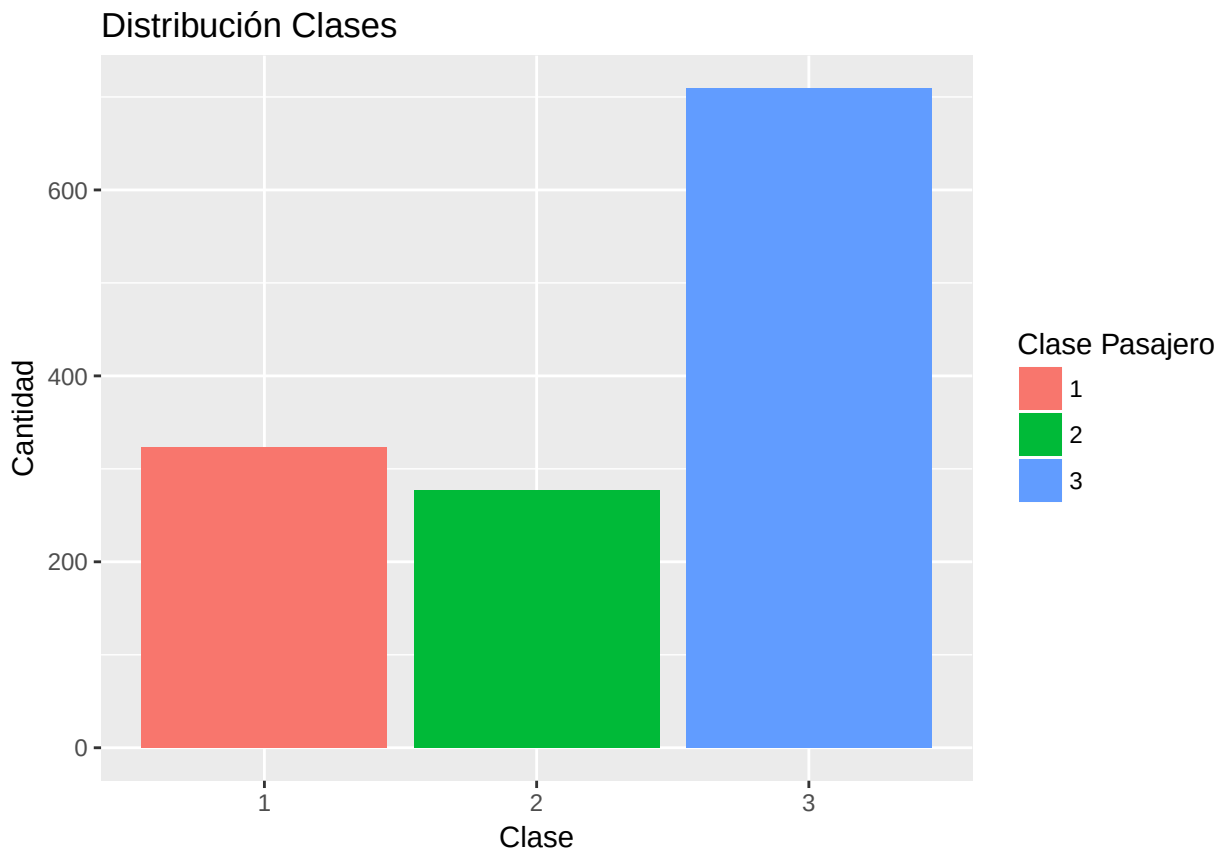
Podemos observar que la media de edad de los pasajeros es de 29.8 años y la mediana es de 28.

Finalmente realizando unos gráficos de las variables principales, clase, sexo y puerto de embarque.

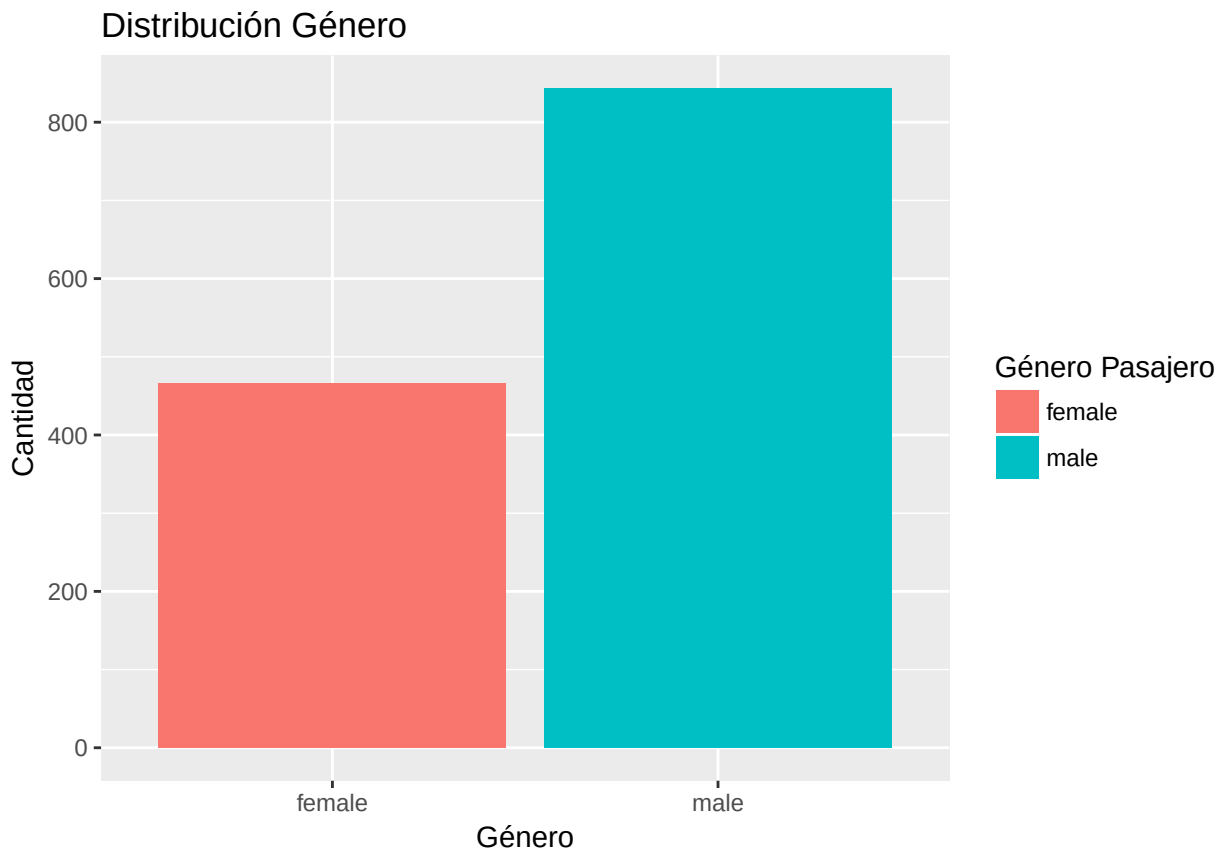
```
par(mfrow=c(1,2))

ggplot(data,  aes(x = Pclass)) +
  geom_histogram(aes(fill = Pclass) ,stat = "count") +
  labs (title = "Distribución Clases", x= "Clase", y = "Cantidad", fill = "Clase Pasajero")
```

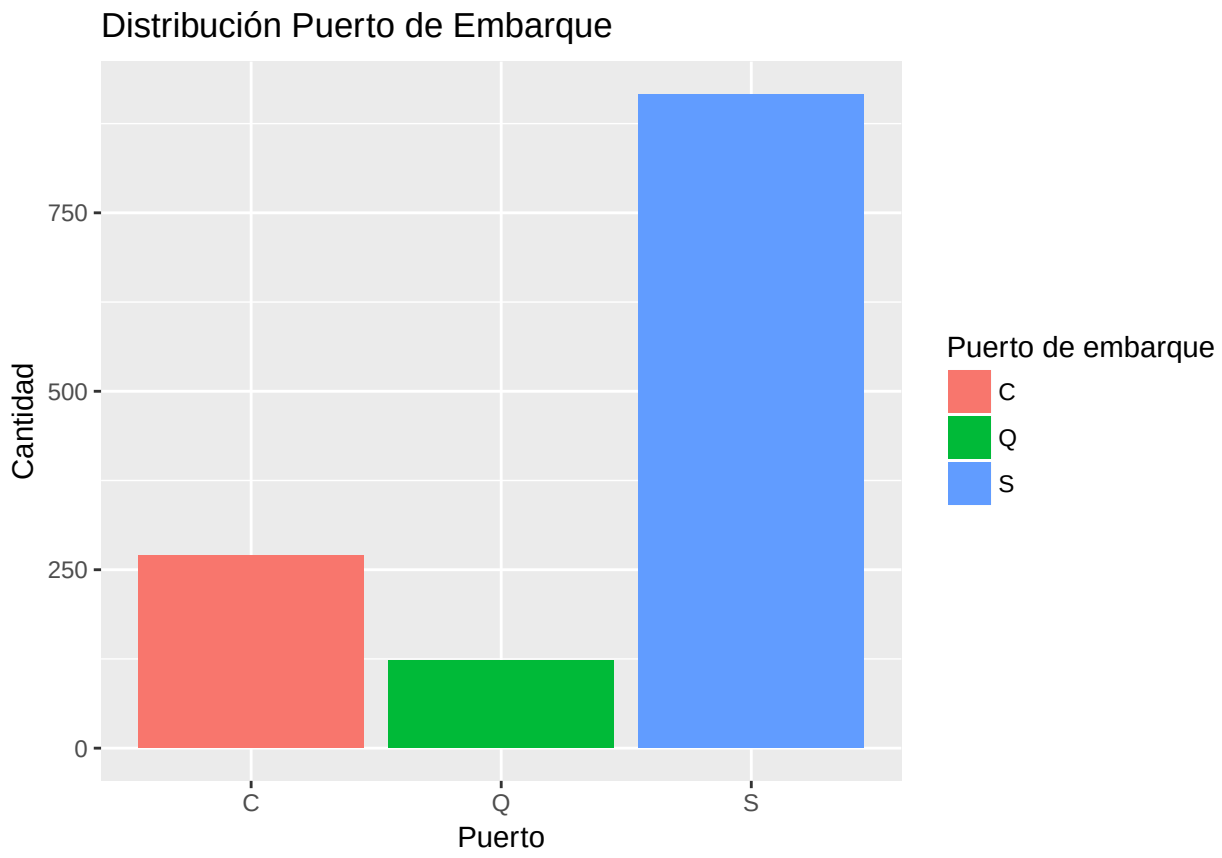
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```




```
qplot(Sex, data = data,  fill= Sex) +
  labs (title = "Distribución Género", x= "Género", y = "Cantidad", fill = "Género Pasajero")
```




```
qplot(Embarked, data = data,  fill= Embarked) +
  labs (title = "Distribución Puerto de Embarque", x= "Puerto",
        y = "Cantidad", fill = "Puerto de embarque")
```



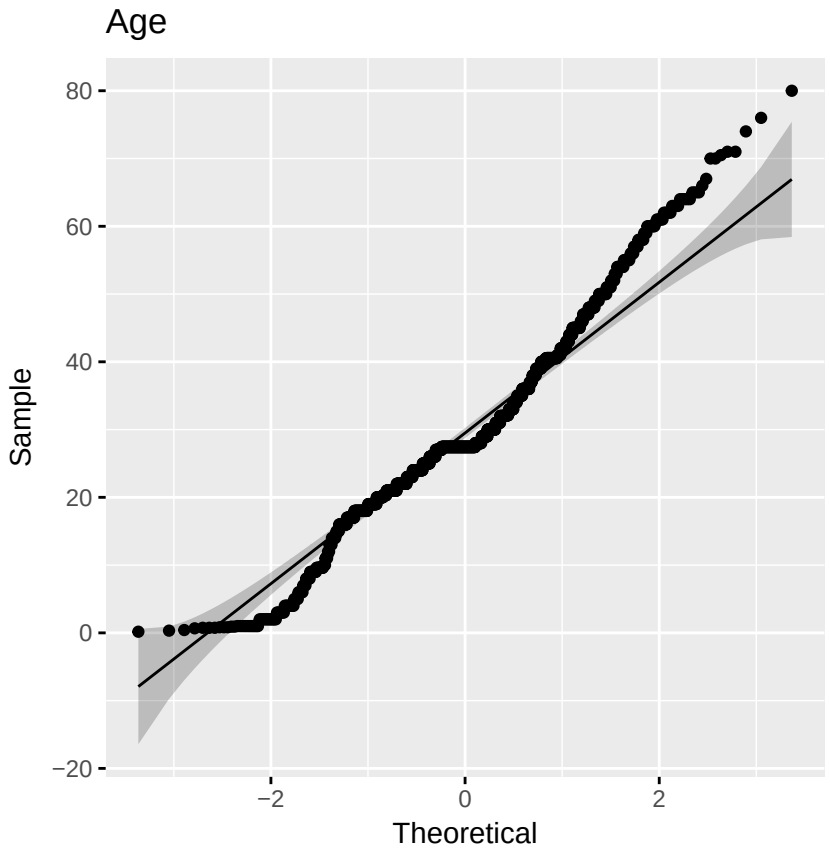
Observamos que la clase más numerosa es la 3ra clase, el sexo predominante es el masculino y el puerto dónde hubo mayor embarque es el de Southampton, al ser el puerto de origen del trasatlántico.

4.2. Comprobación de la normalidad y homogeneidad de la varianza.

Comprobar la normalidad y homogeneidad de la varianza tiene sentido para la variables numéricas Age y Fare.

Comprobamos la normalidad, gráficamente, para Age:

```
#Hago que los dos ejes tengan el mismo tamaño.
ggqqplot(data$Age, ggtheme = theme(aspect.ratio=1), title = "Age")
```



También se puede aplicar un test Shapiro-Wilk, en el que la hipótesis nula, (H0) es que la muestra proviene de una población normalmente distribuida y la hipótesis alternativa (H1), que la muestra no proviene de una población normalmente distribuida.

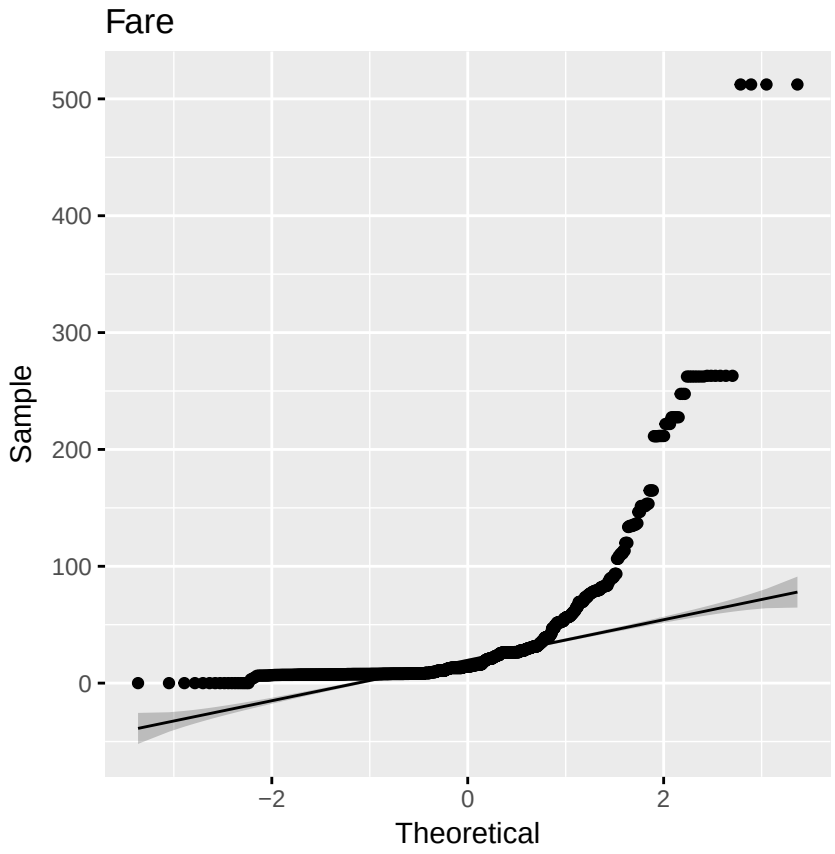
```
# Aplico el test
shapiro.test(data$Age)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data$Age
## W = 0.97034, p-value = 9.628e-16
```

El valor de W está próximo a 1 y el p-value < 0.05, (el p-value, debería ser p-value>0.05 para seguir una distribución normal) así que se rechaza la hipótesis nula y la muestra no sigue una distribución normal. Como se ve en la gráfica hay menos jóvenes y más con elevada edad que habría en una distribución normal.

A continuación, comprobamos la normalidad de Fare, gráficamente:

```
#Hago que los dos ejes tengan el mismo tamaño.
ggqqplot(data$Fare, ggtheme = theme(aspect.ratio=1), title = "Fare")
```



Aplicamos el test de Shapiro-Wilk a la variable Fare

```
# Aplico el test
shapiro.test(data$Fare)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data$Fare
## W = 0.52765, p-value < 2.2e-16
```

En este caso ni siquiera W está cercano a 1, así que se rechaza la hipótesis nula y tampoco sigue una distribución normal.

Comprobamos la homogeneidad de la varianza, dado que hemos visto que los datos no siguen una distribución normal, aplicaremos el test de Fligner-Killeen.

```
# Aplicamos el teest de Fligner-Killeen.
fligner.test(Age ~ Fare, data = data)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Age by Fare
## Fligner-Killeen:med chi-squared = 391.77, df = 280, p-value =
## 1.132e-05
```

Dado que la prueba presenta un p-valor inferior al nivel de significancia (<0.05), se rechaza la hipótesis nula de homocedasticidad y se concluye que la variable Age presenta una varianza estadísticamente diferente para la distribución de Fare.

4.3. Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.

1 - Comprobar la hipotesis que la 1ra clase tiene mas posibilidades de Sobrevivir

H0 : No hay diferencia significativa de sobrevivir entre la clase alta y la clase baja H1 : La clase alta tiene mas probabilidades de sobrevivir

```
Pclass_set <- subset(datostrain, Pclass == 1)

#function for z test
z.test = function(a, b, n){
 sample_mean = mean(a)
 pop_mean = mean(b)
 c = nrow(n)
 var_b = var(b)
 zeta = (sample_mean - pop_mean) / (sqrt(var_b/c))
 return(zeta)
}

#call function
z.test(Pclass_set$Survived, datostrain$Survived, Pclass_set)
```

```
## [1] 7.423828
```

El valor de z de 7.42 afirma la hipotesis alternativa, la clase alta tiene mas probabilidades de sobrevivir.

2 - Comprobación de correlación entre variables

```
chisq.test(data$Sex, data$Pclass)
```

```
##
##  Pearson's Chi-squared test
##
## data:  data$Sex and data$Pclass
## X-squared = 20.379, df = 2, p-value = 3.757e-05
```

Dado que el p valor es menor que 0.05, el genero y la clase son significantes y deben de tomarse en cuenta para realizar cualquier modelo.

3 - Regresion

```
fit <- glm(Survived ~ Age + Pclass + Sex + SibSp + Parch + Fare + Embarked,
           data = datostrain, family = binomial(link = 'logit'))
summary(fit)
```

```
##
## Call:
## glm(formula = Survived ~ Age + Pclass + Sex + SibSp + Parch +
##     Fare + Embarked, family = binomial(link = "logit"), data = datostrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7233  -0.6447  -0.3799   0.6326   2.4457
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.637407   0.634550   8.884  < 2e-16 ***
## Age         -0.043350   0.008232  -5.266 1.39e-07 ***
## Pclass      -1.199251   0.164619  -7.285 3.22e-13 ***
## Sexmale     -2.638476   0.222256 -11.871  < 2e-16 ***
## SibSp       -0.363208   0.129017  -2.815  0.00487 **
## Parch       -0.060270   0.123900  -0.486  0.62666
## Fare         0.001432   0.002531   0.566  0.57165
## EmbarkedQ   -0.823545   0.600229  -1.372  0.17005
## EmbarkedS   -0.401213   0.270283  -1.484  0.13770
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 960.90  on 711  degrees of freedom
## Residual deviance: 632.34  on 703  degrees of freedom
##   (179 observations deleted due to missingness)
## AIC: 650.34
##
## Number of Fisher Scoring iterations: 5
```

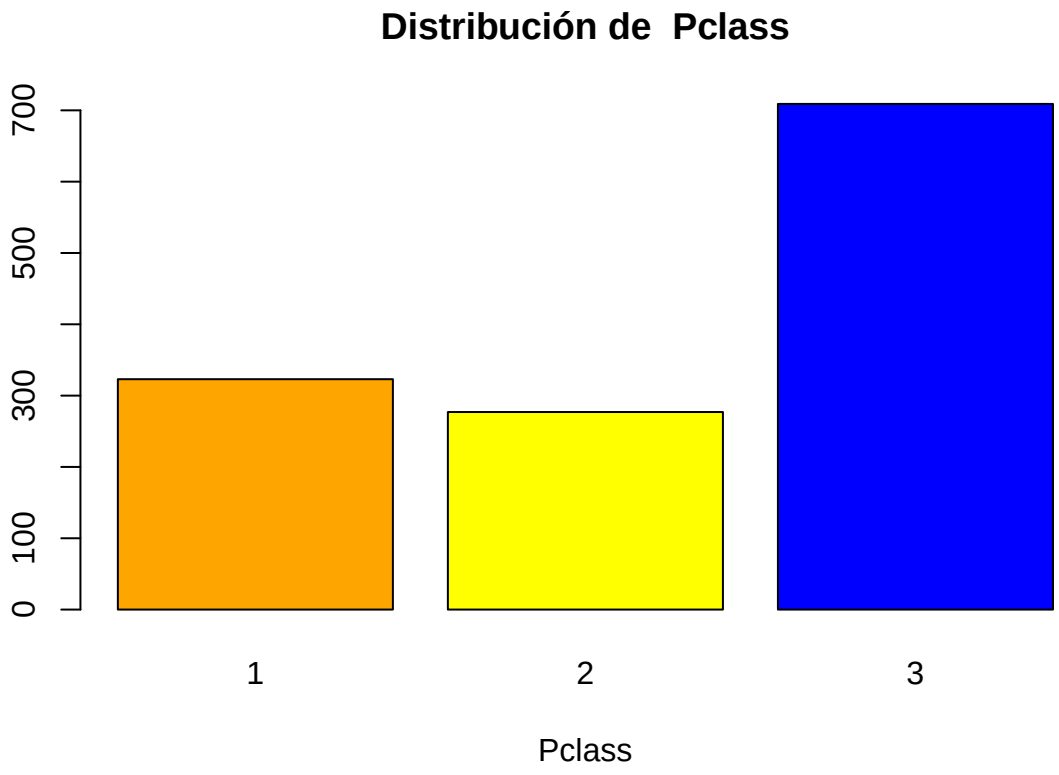
Se comprueba que existe una fuerte relación entre la variable dependiente Survived y Edad, Clase y Genero (hombre).

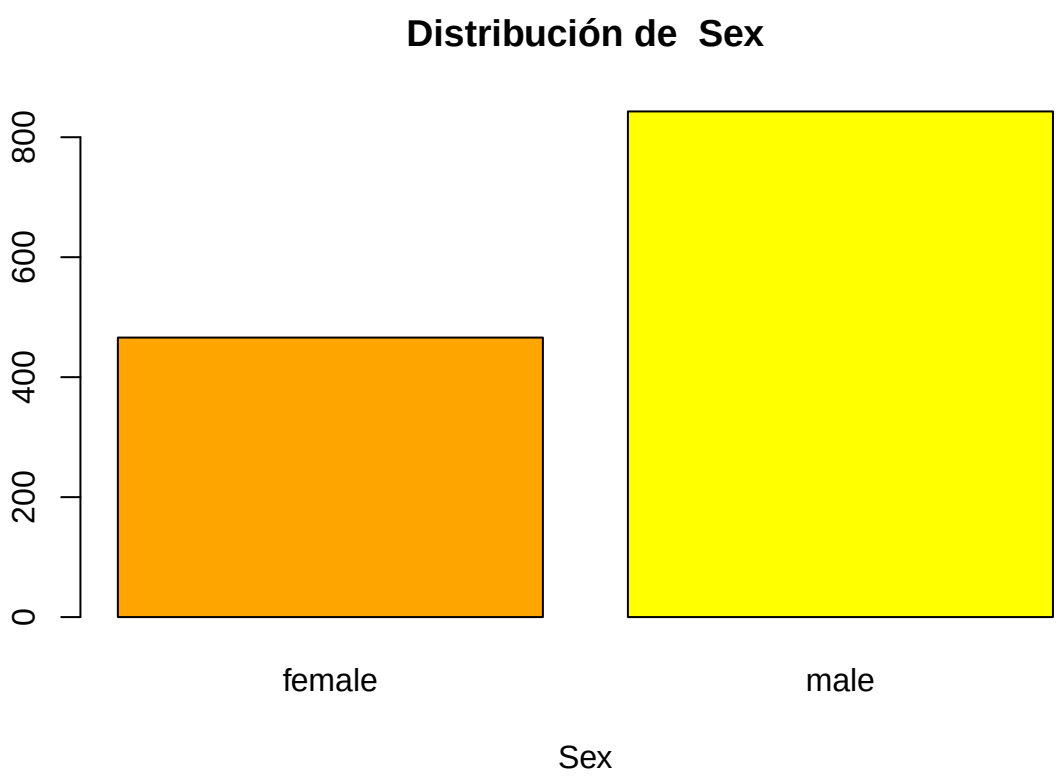
5. Representación de los reultados a apartir de tablas y gráficas.

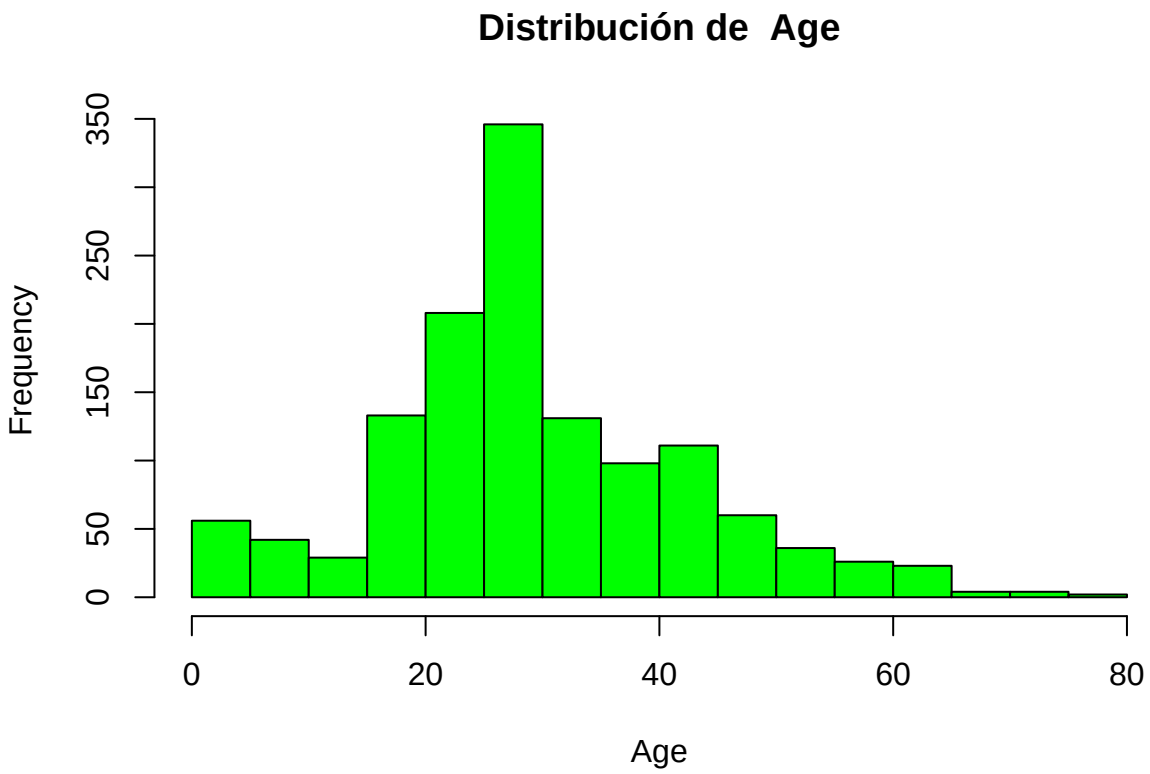
Representamos los datos de las variables.

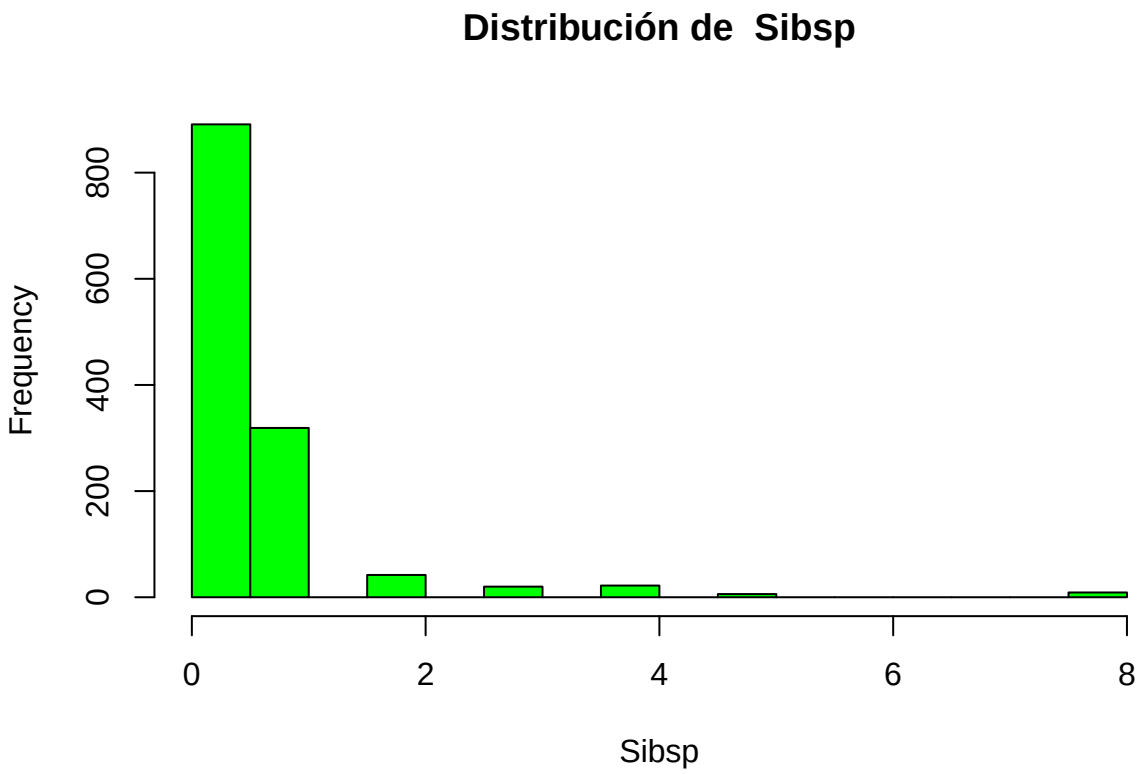
```r
# Mostramos un histograma para cada variable cuantitativa o un gráfico de barras en caso de que
# sea una variable cualitativa.
for (i in 2:ncol(data)) {
  if (class(data[,i]) != "factor" & class(data[,i]) != "character")  {
    hist(data[,i], freq = TRUE, col = c("green"),
         main=paste("Distribución de ",
                    str_to_title(str_replace(colnames(data[i]), "_", " ")), sep = " "),
         xlab= str_to_title(str_replace(colnames(data[i]), "_", " ")))
  }
  else {
    if (class(data[,i]) != "character") {
      barplot(table(data[,i]),
           col = c("orange","yellow","blue","red"),
```

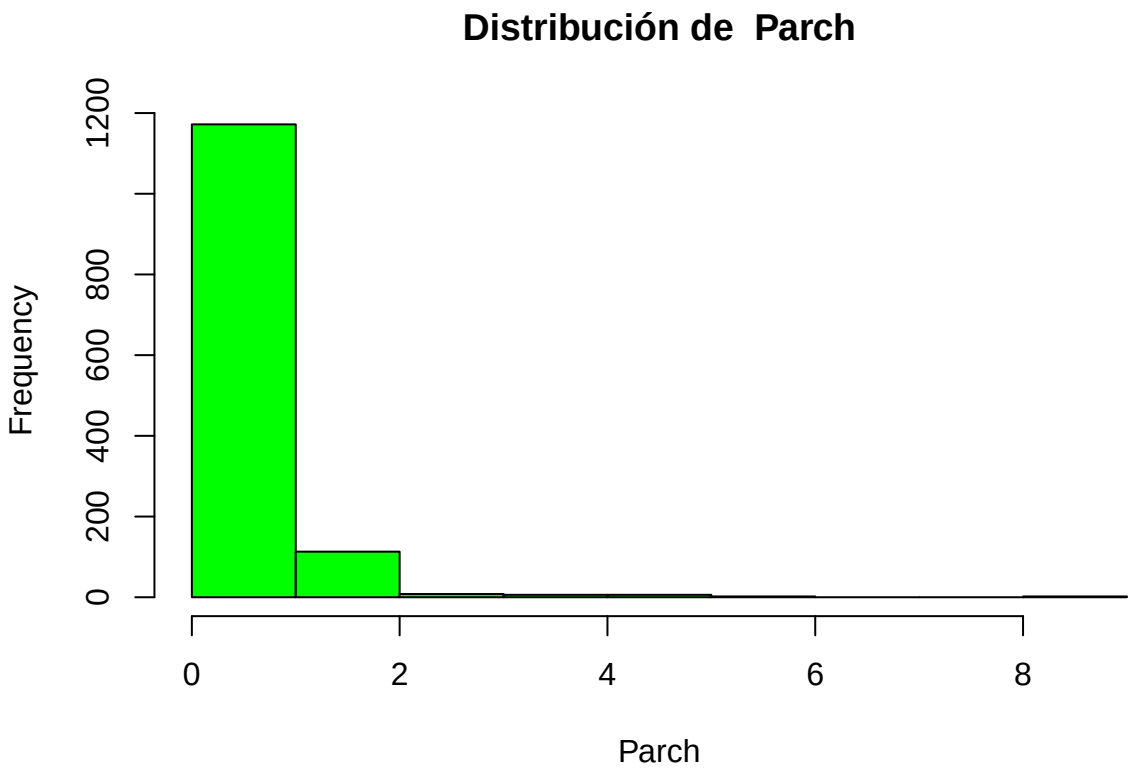
```
            main=paste("Distribución de ",
                   str_to_title(str_replace(colnames(data[i]), "_", " ")), sep = " "),
            xlab= str_to_title(str_replace(colnames(data[i]), "_", " ")))
    }
  }
}
```

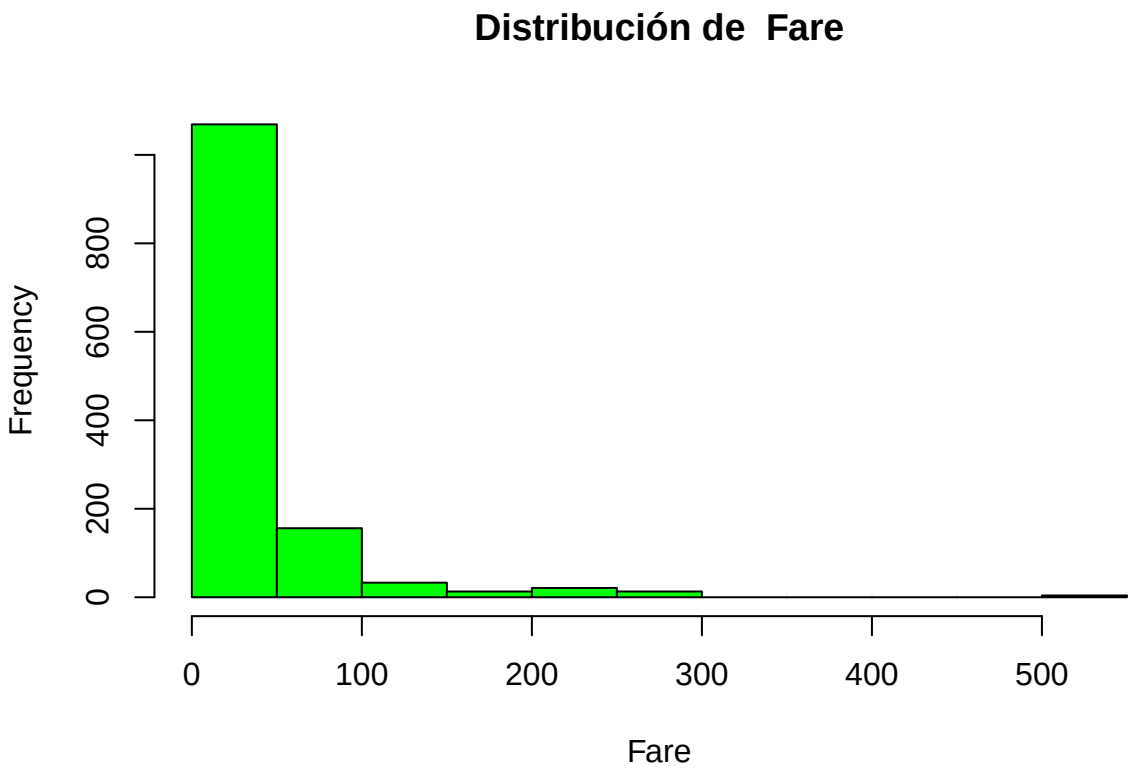


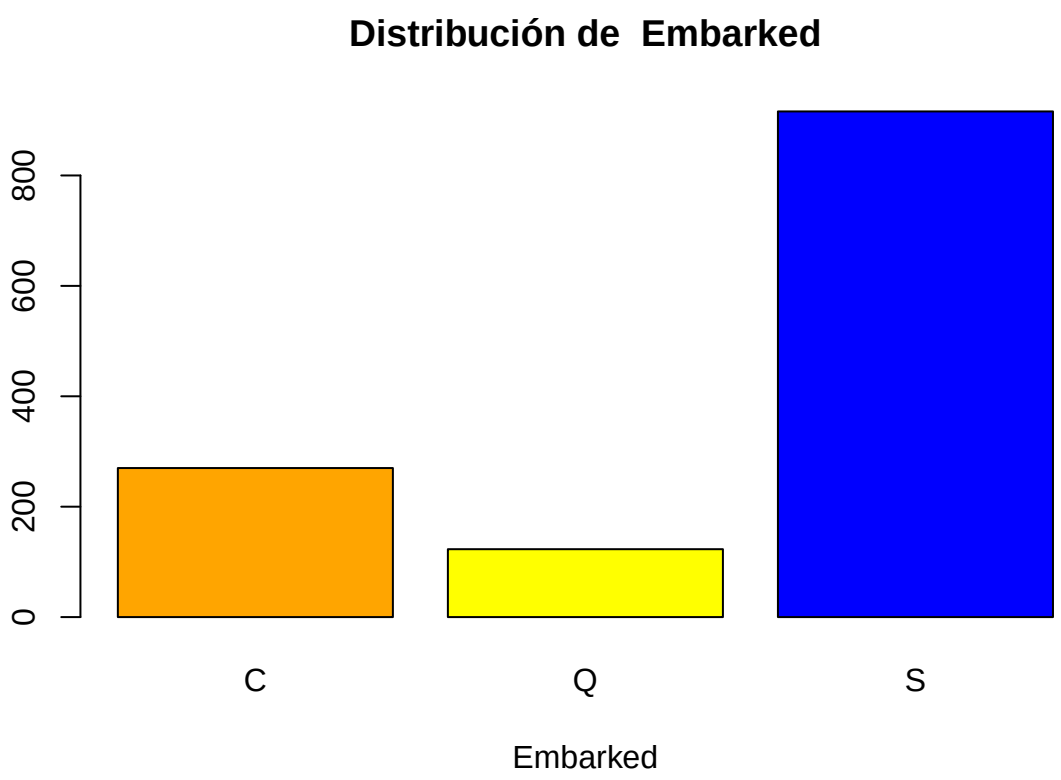

Distribución de Sex
0
200
400
600
800
female
male
Sex

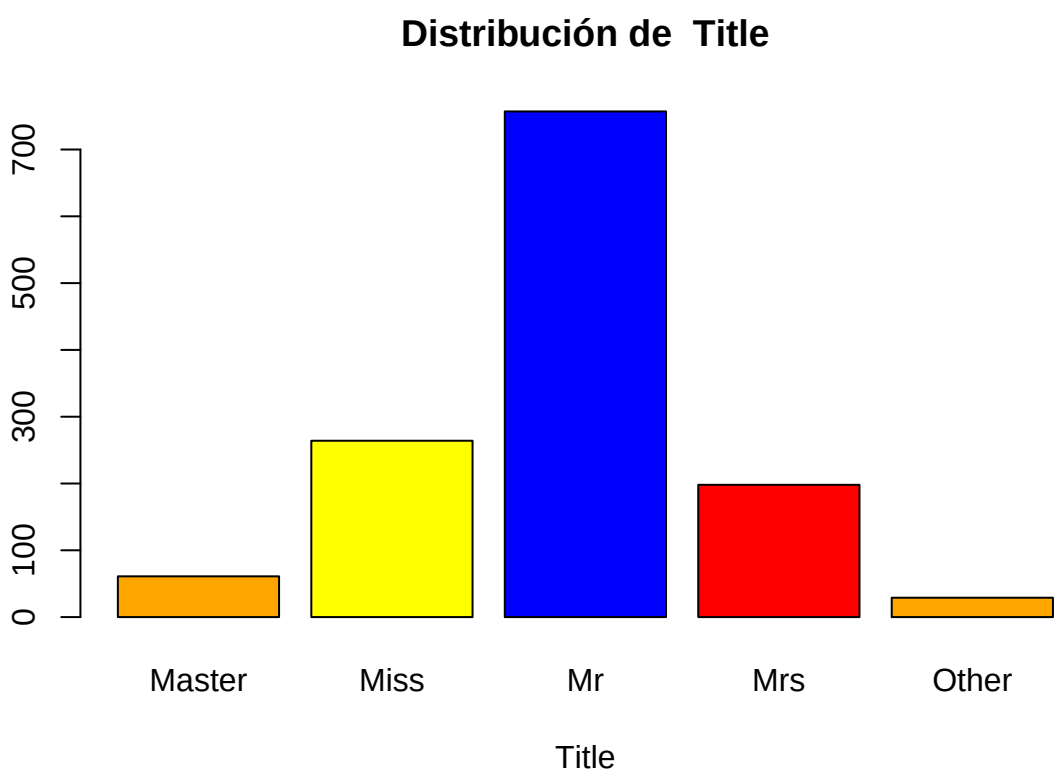

Distribución de Age
Frequency
0
50
150
250
350
0
20
40
60
80
Age

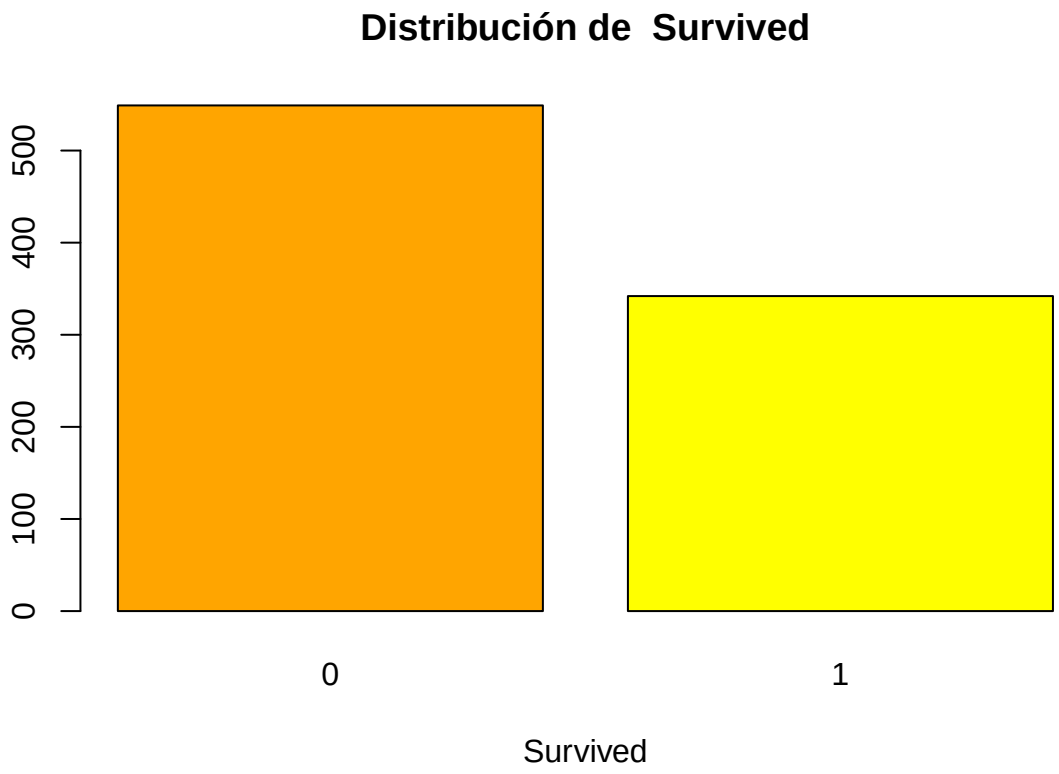

Distribución de Sibsp
Frequency
0
200
400
600
800
0
2
4
6
8
Sibsp


Distribución de Parch
Frequency
0
200
400
600
800
1200
0
2
4
6
8
Parch


Distribución de Fare
Frequency
0
200
400
600
800
0
100
200
300
400
500
Fare


Distribución de Embarked
0
200
400
600
800
C
Q
S
Embarked


Distribución de Title
0
100
300
500
700
Master
Miss
Mr
Mrs
Other
Title

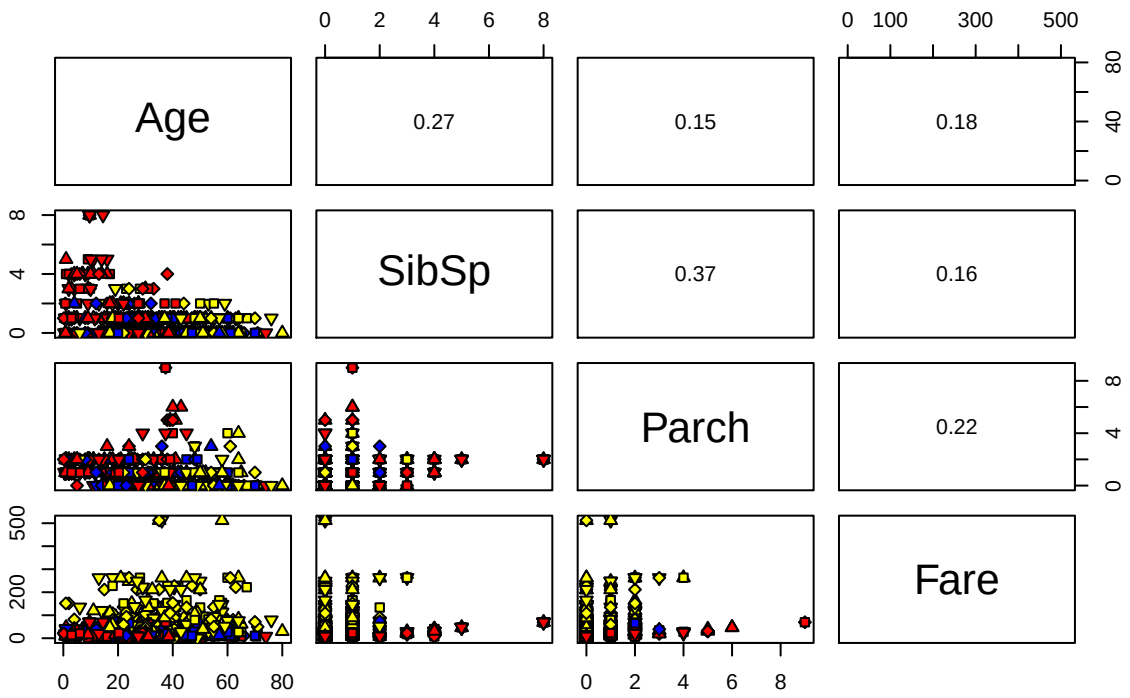


Creamos un gráfico scatter plot, para ver la correlación de las variables:

Hago un gráfico scatterplot para ver la correlación entre variables.

```
# Referencia:
# https://warwick.ac.uk/fac/sci/moac/people/students/peter_cock/r/iris_plots/

# Función para mostrar la correlación
panel.pearson <- function(x, y, ...) {
horizontal <- (par("usr")[1] + par("usr")[2]) / 2;
vertical <- (par("usr")[3] + par("usr")[4]) / 2;
text(horizontal, vertical, format(abs(cor(x,y)), digits=2))
}
# Gráfico de parejas
pairs(data[c(5,6,7,9)], main = "Data plot en función de las clases", pch = c(22,23,24,25),
      bg = c("yellow","blue","red") [unclass(data[,'Pclass'])],
      upper.panel = panel.pearson)
```




```
# Referencia:
# https://warwick.ac.uk/fac/sci/moac/people/students/peter_cock/r/iris_plots/

# Función para mostrar la correlación
panel.pearson <- function(x, y, ...) {
horizontal <- (par("usr")[1] + par("usr")[2]) / 2;
vertical <- (par("usr")[3] + par("usr")[4]) / 2;
text(horizontal, vertical, format(abs(cor(x,y)), digits=2))
}
# Gráfico de parejas
pairs(data[c(5,6,7,9)], main = "Data plot en función del sexo", pch = c(22,23,24,25),
      bg = c("blue","red") [unclass(data[,'Sex'])],
      upper.panel = panel.pearson)
```

6. Resolución del problema. A partir de los resultados obtenidos. ¿cuáles son las conclusiones?. ¿Los resultados permiten responder al problema?

Como se trata de predecir una variable binaria, podemos crear un modelo de regresión logística que sea función de las otras variables.

```
# Referencia
# https://www.kaggle.com/thilakshasilva/predicting-titanic-survival-using-five-algorithms#exploratory-d
# Volvemos a partir el conjunto de datos
titanic_train <- data[1:891, c("Survived","Pclass","Sex","Age","SibSp","Parch","Fare","Embarked","Title
titanic_test <- data[892:1309, c("Pclass","Sex","Age","SibSp","Parch","Fare","Embarked","Title")]
```

Utilizaremos una parte del conjunto de train del que conocemos los resultados, para validar los resultados.

```
# Volvemos a partir el conjunto de datos
set.seed(198)
particion = sample.split(titanic_train$Survived, SplitRatio = 0.8)
train = subset(titanic_train, particion == TRUE)
test = subset(titanic_train, particion == FALSE)
```

```r
# Referencia:
# https://rpubs.com/emilopezcano/logit
# Modelo de regresión logística
titanic.logit <- glm(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked + Title,
                     data = titanic_train, family = "binomial"(link="logit"))
summary(titanic.logit)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age + SibSp + Parch +
##     Fare + Embarked + Title, family = binomial(link = "logit"),
##     data = titanic_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3878  -0.5480  -0.3801   0.5353   2.5499
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  19.665733 504.958265   0.039 0.968934
## Pclass2      -1.130844   0.330031  -3.426 0.000611 ***
## Pclass3      -2.243296   0.331563  -6.766 1.33e-11 ***
## Sexmale     -15.212933 504.957877  -0.030 0.975966
## Age          -0.028446   0.009691  -2.935 0.003331 **
## SibSp        -0.569158   0.126612  -4.495 6.95e-06 ***
## Parch        -0.361322   0.135575  -2.665 0.007697 **
## Fare          0.003362   0.002653   1.267 0.205055
## EmbarkedQ    -0.071733   0.395059  -0.182 0.855916
## EmbarkedS    -0.408573   0.251683  -1.623 0.104512
## TitleMiss   -15.772521 504.958131  -0.031 0.975082
## TitleMr      -3.508822   0.540508  -6.492 8.49e-11 ***
## TitleMrs    -14.972348 504.958197  -0.030 0.976346
## TitleOther   -3.543153   0.781496  -4.534 5.79e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.66  on 890  degrees of freedom
## Residual deviance:  724.73  on 877  degrees of freedom
## AIC: 752.73
##
## Number of Fisher Scoring iterations: 13
```

Vemos como hay variables que no tienen tanta importancia en el clasificador como Sex y Embarked.

Aplicamos la función step para ir quitando estas variables que no tienen tanta importancia.

```r
# Modelo de regresión logística
titanic.logit <- step(titanic.logit)
```

```
## Start:  AIC=752.73
## Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked +
```

```
##     Title
##
##            Df Deviance    AIC
## - Embarked  2   727.94 751.94
## - Fare      1   726.55 752.55
## <none>          724.73 752.73
## - Sex       1   730.19 756.19
## - Parch     1   732.36 758.36
## - Age       1   733.75 759.75
## - SibSp     1   750.31 776.31
## - Pclass    2   774.74 798.74
## - Title     4   782.95 802.95
##
## Step:  AIC=751.94
## Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Title
##
##          Df Deviance    AIC
## <none>       727.94 751.94
## - Fare    1   730.69 752.69
## - Sex     1   733.31 755.31
## - Parch   1   736.27 758.27
## - Age     1   737.68 759.68
## - SibSp   1   756.65 778.65
## - Pclass  2   780.00 800.00
## - Title   4   786.70 802.70
```

```r
# Modelo de regresión logística
summary(titanic.logit)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age + SibSp + Parch +
##     Fare + Title, family = binomial(link = "logit"), data = titanic_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4621  -0.5405  -0.3886   0.5325   2.6471
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  19.410542 498.876533   0.039 0.968963
## Pclass2      -1.223069   0.324793  -3.766 0.000166 ***
## Pclass3      -2.248701   0.321985  -6.984 2.87e-12 ***
## Sexmale     -15.189027 498.876155  -0.030 0.975711
## Age          -0.029122   0.009569  -3.043 0.002340 **
## SibSp        -0.593281   0.125619  -4.723 2.33e-06 ***
## Parch        -0.373677   0.134520  -2.778 0.005472 **
## Fare          0.004043   0.002633   1.535 0.124694
## TitleMiss   -15.741626 498.876416  -0.032 0.974828
## TitleMr      -3.554349   0.537396  -6.614 3.74e-11 ***
## TitleMrs    -14.973049 498.876481  -0.030 0.976056
## TitleOther   -3.514816   0.775871  -4.530 5.89e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.66  on 890  degrees of freedom
## Residual deviance:  727.94  on 879  degrees of freedom
## AIC: 751.94
##
## Number of Fisher Scoring iterations: 13
```

```
# Modelo de regresión logística
titanic.logit <- step(titanic.logit)
```

```
## Start:  AIC=751.94
## Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Title
##
##          Df Deviance    AIC
## <none>        727.94 751.94
## - Fare    1   730.69 752.69
## - Sex     1   733.31 755.31
## - Parch   1   736.27 758.27
## - Age     1   737.68 759.68
## - SibSp   1   756.65 778.65
## - Pclass  2   780.00 800.00
## - Title   4   786.70 802.70
```

```
# Modelo de regresión logística
vif(titanic.logit)
```

```
##                GVIF Df GVIF^(1/(2*Df))
## Pclass 2.169085e+00  2        1.213582
## Sex    6.818832e+06  1     2611.289253
## Age    1.991458e+00  1        1.411190
## SibSp  1.614081e+00  1        1.270465
## Parch  1.458854e+00  1        1.207830
## Fare   1.586248e+00  1        1.259464
## Title  1.492792e+07  4        7.884060
```

```
# Modelo de regresión logística
durbinWatsonTest(titanic.logit)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.0204161      1.958783   0.568
##  Alternative hypothesis: rho != 0
```

```
# Comprobamos los resultados en el conjunto de test de validación
prob_pred = predict(titanic.logit, type = 'response', newdata = test)
y_pred = ifelse(prob_pred > 0.5, 1, 0)
head(y_pred)
```

```
##  1  2 12 13 16 22
##  0  1  1  0  1  0
```

Comprobamos la matriz de confusion

```
# Comprobamos la matriz de confusión
table(test$Survived, y_pred > 0.5)
```

```
##
##     FALSE TRUE
##   0    97   13
##   1    17   51
```

```
#kable(table(test$Survived, (y_pred > 0.5)) %>%
#  kable_styling(bootstrap_options = "striped", full_width = F, position = "left")
```

Calculamos la precisión del modelo.

```
error <- mean(test$Survived != y_pred) # Misclassification error
paste('Accuracy',round(1-error,4))
```

```
## [1] "Accuracy 0.8315"
```

Finalmente, calculamos los valores finales

```
# Calculamos las predicciones
titanic_prob = predict(titanic.logit, newdata = titanic_test)
titanic_pred = ifelse(titanic_prob > 0.5, 1, 0)
```

```
# Guardamos los resultados
results <- data.frame(PassengerID = data[892:1309,"PassengerId"], Survived = titanic_pred)
```

```
# Guardamos el archivo
write.csv(results, file = 'PrediccionSupervivenciaTitanic.csv', row.names = FALSE, quote=FALSE)
```

7. Código. Hay que adjuntar el código, preferiblemente en R, con el que se ha realizado la limpieza, análisis y represntación de los datos.